DS1 : Révisions de logique

Documents et calculatrice interdits. Le sujet comporte 4 pages et est à traiter en 3 heures. Il est constitué de deux problèmes indépendants pouvant être traités dans un ordre quelconque. Le devoir sera noté sur 30 points, le barème est donné à titre indicatif et est suceptible d'être modifié.

Problème 1 *Logiques trivaluées (sur 14 points)*

Dans tout l'énoncé, $\mathcal{F}$ désigne l'ensemble des formules du calcul propositionnel n'utilisant que les connecteurs de $\mathcal{C} = \{\vee, \wedge, \neg, \Rightarrow\}$ construites sur un ensemble $\mathcal{V}$ de variables propositionnelles. La sémantique standard du calcul propositionnel, qu'on notera S_2 par la suite, ne fait intervenir que deux valeurs de vérité : vrai (noté V) et faux (noté F). Dans cet énoncé, on cherche à étendre cette sémantique bivaluée classique de sorte à prendre en compte une troisième valeur de vérité, notée I (pour indéterminé).

Se donner une sémantique trivaluée revient à donner les tables de vérité des connecteurs de $\mathcal{C}$. Si S_3 est une sémantique trivaluée, les concepts sémantiques introduits pour S_2 s'étendent à S_3 comme suit :

- Une *trivaluation* est une fonction $v : \mathcal{V} \cup \{\bot, \top\} \to \{\mathrm{V}, \mathrm{F}, \mathrm{I}\}$ telle que $v(\top) = \mathrm{V}$ et $v(\bot) = \mathrm{F}$. Si v est une trivaluation et G une formule, $v(G) \in \{\mathrm{V}, \mathrm{I}, \mathrm{F}\}$ est définie par induction à l'aide des tables de vérité des connecteurs de manière similaire au cas où la sémantique est S_2.
- Une formule $G \in \mathcal{F}$ est *satisfiable* (selon S_3) s'il existe une trivaluation v telle que $v(G) = \mathrm{V}$. Une formule $G \in \mathcal{F}$ est une *tautologie* (selon S_3) si pour toute trivaluation v on a $v(G) = \mathrm{V}$.
- Deux formules G et H sont *équivalentes* (selon S_3) si pour toute trivaluation v on a $v(G) = v(H)$.
- Si Γ est un ensemble de formules et G une formule, G est *conséquence sémantique* de Γ (selon S_3), noté $\Gamma \models G$, si toute trivaluation satisfaisant Γ satisfait aussi G.

Dans un premier temps, on considère la sémantique de Łukasiewicz, notée S_L : c'est une sémantique trivaluée dans laquelle la sémantique des connecteurs $\neg, \vee, \wedge$ et $\Rightarrow$ est définie par les tables de vérité suivantes :

x	$\neg x$
V	F
F	V
I	I

x	y	$x \wedge y$
V	V	V
V	F	F
V	I	I
F	V	F
F	F	F
F	I	F
I	V	I
I	F	F
I	I	I

x	y	$x \vee y$
V	V	V
V	F	V
V	I	V
F	V	V
F	F	F
F	I	I
I	V	V
I	F	I
I	I	I

x	y	$x \Rightarrow y$
V	V	V
V	F	F
V	I	I
F	V	V
F	F	V
F	I	V
I	V	V
I	F	I
I	I	V

1. Etablir soigneusement la table de vérité selon S_L de la formule $((A \Rightarrow B) \wedge (\neg A \Rightarrow B)) \Rightarrow B$. Est-elle satisfiable ? Est-elle une tautologie ?
2. La formule $A \vee \neg A$ est-elle une tautologie dans S_L ? Commenter.
3. Donner une formule qui est tautologique dans la sémantique S_L.

4. Indiquer une formule G_1 équivalente selon S_2 à la formule $A \vee B$ et ne faisant intervenir que les connecteurs $\neg$ et $\wedge$. Les formules G_1 et $A \vee B$ sont-elles équivalentes selon S_L ?

5. Indiquer une formule G_2 équivalente selon S_2 à la formule $A \Rightarrow B$ et ne faisant intervenir que les connecteurs $\neg$ et $\vee$. Les formules G_2 et $A \Rightarrow B$ sont-elles équivalentes selon S_L ?

6. Est-il possible de raisonner par contraposition dans S_L ? Justifier.

On considère dans la question suivante une nouvelle sémantique trivaluée S_I définie comme suit : la sémantique des connecteurs $\neg, \vee$ et $\wedge$ prolonge celle de S_2 en attribuant la valeur I dès qu'un des opérandes de ces connecteurs vaut I. La sémantique de $\Rightarrow$ est la même que dans S_L.

7. On note $(\star)$ l'affirmation suivante :

   "Pour toutes $G, H \in \mathcal{F}$, on a $G \models H$ si et seulement si $\models G \Rightarrow H$".

   a) Prouver que l'affirmation $(\star)$ est vraie selon la sémantique standard S_2.

   b) Reste-t-elle vraie selon la sémantique S_I ? Justifier.

   c) Si la réponse à la question précédente est négative, proposer une nouvelle sémantique qui, par rapport à S_I, ne modifie que la sémantique pour le connecteur $\Rightarrow$ et dans laquelle $(\star)$ est vraie. On justifiera que $(\star)$ est bien vérifiée dans la sémantique proposée.

On considère à présent la sémantique de Kleene et Priest, notée S_K. Dans cette sémantique, les tables de vérité de $\neg, \vee$ et $\wedge$ sont identiques à celles dans S_L, seule change la table de vérité de $\Rightarrow$ (à la dernière ligne) qui devient la suivante :

x	y	$x \Rightarrow y$
V	V	V
V	F	F
V	I	I
F	V	V
F	F	V
F	I	V
I	V	V
I	F	I
I	I	I

8. Est-il vrai que pour toute formule A, la formule $A \Rightarrow A$ est une tautologie dans S_K ?

9. On note $\mathcal{F}_v$ l'ensemble des "vraies formules", défini par induction de la même façon que $\mathcal{F}$ à l'exception près que ni $\top$ ni $\bot$ ne sont des cas de base (autrement dit, ni $\top$ ni $\bot$ ni aucune formule de $\mathcal{F}$ construite avec $\top$ ou $\bot$ n'appartiennent à $\mathcal{F}_v$). Montrer qu'aucune formule de $\mathcal{F}_v$ n'est une tautologie selon la sémantique S_K.

Les questions précédentes proposent différentes sémantiques trivaluées se comportant plus ou moins différemment de S_2. On cherche à présent à dénombrer les sémantiques trivaluées satisfaisant un certain nombre de critères garantissant une certaine proximité avec les propriétés de la sémantique bivaluée standard. On définit un ordre sur les valeurs de vérité F, I et V : F $<$ I $<$ V.

10. On considère une sémantique trivaluée S vérifiant simultanément :

    (1) S étend S_2 ; autrement dit, restreindre les tables de vérité selon S de $\neg, \vee, \wedge, \Rightarrow$ à V et F donne les tables de vérité de ces connecteurs selon S_2.

    (2) Le connecteur $\wedge$ est commutatif.

(3) La formule $A \vee B$ est équivalente à G_1 (définie en question 4).

(4) La formule $A \Rightarrow B$ est équivalente à G_2 (définie en question 5).

(5) Les formules $\neg\neg A$ et A sont équivalentes.

(6) Pour toute valeur de vérité y, la fonction $\begin{cases} \{\mathrm{V}, \mathrm{I}, \mathrm{F}\} \to \{\mathrm{V}, \mathrm{I}, \mathrm{F}\} \\ x \mapsto x \wedge y \end{cases}$ est croissante au sens large.

Combien y a-t-il de telles sémantiques ? Justifier.

Problème 2 *HORN-SAT (sur 16 points)*

On s'intéresse dans ce problème à la satisfiabilité de formules sous forme normale conjonctive particulières : les formules de Horn. Ces dernières sont définies comme suit :

Un littéral est une variable propositionnelle ou sa négation. Le littéral est dit positif dans le premier cas et négatif dans le second. Une *clause de Horn* est une disjonction de littéraux distincts dont au plus un est positif. Une *formule de Horn* est une conjonction de $n \geq 0$ clauses de Horn.

1. Indiquer parmi les formules de Horn suivantes lesquelles sont satisfiables en justifiant brièvement :
   a) $H_1 = (\neg v_0 \vee v_1) \wedge (v_0 \vee \neg v_1 \vee \neg v_2)$.
   b) $H_2 = v_1 \wedge (\neg v_0 \vee \neg v_2) \wedge \neg v_1 \wedge (v_0 \vee \neg v_2 \vee \neg v_3)$.
   c) $H_3 = v_1 \wedge (\neg v_0 \vee \neg v_1) \wedge (v_0 \vee \neg v_1) \wedge (v_0 \vee \neg v_1 \vee \neg v_2)$
2. Soit H une formule de Horn dont chacune des clauses contient au moins un littéral négatif. Montrer que H est satisfiable en exhibant une valuation adéquate.
3. Soit H une formule de Horn dont une des clauses est réduite au littéral positif v. Si aucune clause de H n'est réduite à $\neg v$, montrer en justifiant qu'on peut construire à partir de H une formule de Horn H' telle que H est satisfiable si et seulement si H' l'est et telle que H' ne fait plus intervenir la variable v.
4. En déduire le pseudo-code d'un algorithme permettant de vérifier la satisfiabilité d'une formule de Horn. Expliquer brièvement sa terminaison et sa correction.
5. Etudier la complexité pire cas de l'algorithme proposé en 4 sur une formule de Horn H en fonction du nombre n de variables propositionnelles dans H et de son nombre de clauses m. On discutera au besoin des structures de données que l'on pourrait utiliser pour implémenter cet algorithme.
6. Le problème HORN-SAT est le suivant :

   Entrée : Une formule de Horn H.
   Sortie : Oui si H est satisfiable, non sinon.

   Que peut-on dire du problème HORN-SAT par rapport au problème SAT ?

Les questions suivantes visent à implémenter en Ocaml un algorithme prenant en entrée une formule de Horn et calculant une valuation la satisfaisant, si elle existe. Pour construire une telle valuation, on s'inspire des idées des questions précédentes et on procède comme suit :

(1) On initialise une valuation v attribuant faux à toutes les variables de la formule H. Cette valuation sera modifiée petit à petit jusqu'à ce qu'elle satisfasse F si F est effectivement satisfiable.

(2) On cherche une clause c de H non satisfaite par v : s'il n'y en a pas, v satisfait H.

(3) Sinon on vérifie si c contient un littéral positif p. Si oui, on modifie v de sorte à attribuer la valeur vrai à p et on repart au point (2). Sinon, on conclut à la non satisfiabilité de H.

7. Montrer que la clause de Horn $C = (v \vee \neg v_0 \vee ... \vee \neg v_k)$ est équivalente à l'implication $(v_0 \wedge ... \wedge v_k) \Rightarrow v$. Cette propriété explique pourquoi on appelle les littéraux négatifs d'une clause de Horn ses hypothèses et son éventuel littéral positif sa conséquence.

On introduit les types Ocaml suivants afin de rendre compte de formules de Horn :

```
type variable = int
type hornclause = {csqc : variable option ; hyp : variable list}
type hornformule = int* hornclause list
type valuation = bool array
```

Une variable est représentée par un entier (correspondant intuitivement à son indice). On supposera que dans toutes les formules manipulées, les variables utilisées sont numérotées de 0 à $k-1$ où k est le nombre de variables distinctes intervenant dans la formule.

Une clause de Horn est représentée par un couple constitué de son éventuelle conclusion (la variable de l'éventuel littéral positif) et de la liste de ses hypothèses (les variables de ses littéraux négatifs). Une formule de Horn est représentée par une liste de clauses, à laquelle on adjoint le nombre de variables distinctes qui y interviennent. Une valuation est représentée par un tableau contenant en case i la valeur de vérité attribuée à la variable (d'indice) i. Par exemple, la formule H_3 est représentée par :

```
let h3 = (3,[{csqc = Some 1; hyp = []};
             {csqc = None; hyp = [0;1]};
             {csqc = Some 0; hyp = [1]};
             {csqc = Some 0; hyp = [1;2]}])
```

On rappelle que l'accès au champ csqc (par exemple) d'une variable hc de type hornclause se fait à l'aide de la syntaxe hc.csqc.

8. Ecrire une fonction evalue de signature hornclause -> valuation -> bool telle que evalue c v renvoie true si v satisfait la clause c et false sinon.

9. Ecrire une fonction trouve_clause_fausse de signature hornclause list -> valuation -> hornclause option telle que trouve_clause_fausse lc v renvoie None si aucune clause de lc n'est rendue fausse par v et Some c où c est une clause de lc non satisfaite par v sinon.

10. Déduire des questions précédentes une fonction hornsat de signature hornformule -> valuation option telle que hornsat f renvoie None si f n'est pas satisfiable et Some val où val est une valuation satisfaisant f sinon. On expliquera soigneusement le fonctionnement de la fonction proposée.

11. Etudier la complexité temporelle pire cas de hornsat en fonction de grandeurs pertinentes.